Package 'ClusTorus'

July 26, 2021

Type Package

Title Prediction and Clustering on the Torus by Conformal Prediction

Description Provides various tools of for clustering multivariate angular data on the torus. The package provides angular adaptations of usual clustering methods such as the k-means clustering, pairwise angular distances, which can be used as an input for distance-based clustering algorithms, and implements clustering based on the conformal prediction framework. Options for the conformal scores include scores based on a kernel density estimate, multivariate von Mises mixtures, and naive k-means clusters. Moreover, the package provides some basic data handling tools for angular data.

Version 0.1.3

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

URL https://github.com/sungkyujung/ClusTorus

BugReports https://github.com/sungkyujung/ClusTorus/issues

Depends R (>= 3.6.0)

Imports BAMBI, igraph, purrr, ggplot2, rlang, stats, utils

Suggests knitr, rmarkdown, tidyverse, cowplot

VignetteBuilder knitr

NeedsCompilation no

Author Sungkyu Jung [aut, cph],
Seungki Hong [aut, cre],
Kiho Park [ctb],
Byungwon Kim [ctb]

Maintainer Seungki Hong <skgaboja@snu.ac.kr>

Repository CRAN

Date/Publication 2021-07-26 09:10:02 UTC

R topics documented:

`ang.dist` *Angular distance*

Description

`ang.dist` computes element-wise angular distance between two angular values in $[0, 2\pi)$.

Usage

```
ang.dist(x, y)
```

Arguments

`x, y`	angular data(both scalar or vector) whose elements are in $[0, 2\pi)$

Value

angular data (scalar or vector) whose elements are in $[0, 2\pi)$

References

S. Jung, K. Park, and B. Kim (2021), "Clustering on the torus by conformal prediction"

Examples

```
x <- c(pi/3, 0)
y <- c(pi/4, pi/2)

ang.dist(x, y)
```

`ang.minus` *Angular subtraction*

Description

`ang.minus` computes element-wise angular subtraction defined as

$$x - y := Arg(exp(i(x - y)))$$

Usage

```
ang.minus(x, y)
```

Arguments

`x, y`	angular data(scalar or vector) whose elements are in $[0, 2\pi)$

Value

returns a scalar or a vector whose elements are in $[-\pi, \pi)$.

References

S. Jung, K. Park, and B. Kim (2021), "Clustering on the torus by conformal prediction"

Examples

```
x <- c(pi/2, 0)
y <- c(pi, pi/3)

ang.minus(x, y)
```

ang.pdist *Pairwise L2 angular distance*

Description

ang.pdist computes pairwise angular distances matrix.

Usage

```
ang.pdist(data)
```

Arguments

data	n x d angular data on $[0, 2\pi)^d$

Value

ang.pdist returns pairwise angular distances matrix with the class dist

References

S. Jung, K. Park, and B. Kim (2021), "Clustering on the torus by conformal prediction"

See Also

ang.dist

Examples

```
data <- matrix(c(pi/3, pi/3, pi/2,
                 pi, pi/4, pi/2,
                 0, pi/3, pi/6),
               ncol = 3, byrow = TRUE)

ang.pdist(data)
```

cluster.assign.torus *Clustering by connected components of ellipsoids*

Description

cluster.assign.torus returns clustering assignment for data given icp.torus objects, which can be constructed with icp.torus.score.

Usage

```
cluster.assign.torus(
  data,
  icp.torus,
  level = 0.1,
  intersection.plot = TRUE,
  coord = NULL
)
```

Arguments

`data`	n x d matrix of toroidal data on $[0, 2\pi)^d$.
`icp.torus`	an object containing all values to compute the conformity score, which will be constructed with `icp.torus.score`.
`level`	a scalar in $[0, 1]$. Default value is 0.1.
`intersection.plot`	boolean index. If `TRUE`, then plot the intersections of given ellipsoids. Default is `TRUE`.
`coord`	a 2-vector for prespecifing the coordinates. Default value is `NULL` and automatically generates all combinations of coordinates.

Value

clustering assignment for data, given `icp.torus` objects

References

S. Jung, K. Park, and B. Kim (2021), "Clustering on the torus by conformal prediction",

I. Gilitschenski and U. D. Hanebeck, "A robust computational test for overlap of two arbitrary-dimensional ellipsoids in fault-detection of Kalman filters"

See Also

`icp.torus.score`

Examples

```
data <- toydata1[, 1:2]
icp.torus <- icp.torus.score(data, method = "kmeans",
                             kmeansfitmethod = "general",
                             param = list(J = 4, concentration = 25))
level <- 0.1

cluster.assign.torus(data, icp.torus, level)
```

cp.torus.kde *Conformal prediction set indices with kernel density estimation*

Description

cp.torus.kde computes conformal prediction set indices (TRUE if in the set) using kernel density estimation as conformity score.

Usage

```
cp.torus.kde(data, eval.point = grid.torus(), level = 0.1, concentration = 25)
```

Arguments

data	n x d matrix of toroidal data on $[0, 2\pi)^d$
eval.point	N x N numeric matrix on $[0, 2\pi)^d$. Default input is NULL, which represents the fine grid points on $[0, 2\pi)^d$.
level	either a scalar or a vector, or even NULL. Default value is 0.1.
concentration	positive number which has the role of κ of von Mises distribution. Default value is 25.

Value

If level is NULL, then return kde at eval.point and at data points.

If level is a vector, return the above and prediction set indices for each value of level.

References

S. Jung, K. Park, and B. Kim (2021), "Clustering on the torus by conformal prediction"

See Also

kde.torus, grid.torus

Examples

```
data <- ILE[1:200, 1:2]
cp.torus.kde(data, eval.point = grid.torus(),
             level = 0.05, concentration = 25)
```

EMsinvMmix *Fitting mixtures of bivariate von Mises distribution*

Description

EMsinvMmix returns fitted parameters of J-mixture of bivariate sine von Mises.

Usage

```
EMsinvMmix(
  data,
  J = 4,
  parammat = EMsinvMmix.init(data, J),
  THRESHOLD = 1e-10,
  maxiter = 200,
  type = c("circular", "axis-aligned", "general"),
  kmax = 500,
  verbose = TRUE
)
```

Arguments

data	n x 2 matrix of toroidal data on $[0, 2\pi)^2$
J	number of components of mixture density
parammat	6 x J parameter data with the following components: parammat[1,] : the weights for each von Mises sine density parammat[n + 1,] : κ_n for each von Mises sine density for n = 1, 2, 3 parammat[m + 4,] : μ_m for each von Mises sine density for m = 1, 2
THRESHOLD	number of threshold for difference between updating and updated parameters.
maxiter	the maximal number of iteration.
type	a string one of "circular", "axis-aligned", "general", and "Bayesian" which determines the fitting method.
kmax	the maximal number of kappa. If estimated kappa is larger than kmax, then put kappa as kmax.
verbose	boolean index, which indicates whether display additional details as to what the algorithm is doing or how many loops are done.

Details

This algorithm is based on ECME algorithm. That is, constructed with E - step and M - step and M - step maximizes the parameters with given type.

If type == "circular", then the mixture density is just a product of two independent von Mises.

If type == "axis-aligned", then the mixture density is the special case of type == "circular": only need to take care of the common concentration parameter.

Iftype == "general", then the fitting the mixture density is more complicated than before, check the detail of the reference article.

Value

returns approximated parameters for bivariate normal distribution with `list`:

`list$Sigmainv[j]` : approximated covariance matrix for j-th bivariate normal distribution, approximation of the j-th von Mises.

`list$c[j]` : approximated $|2\pi\Sigma|^{-1}$ for j-th bivariate normal distribution, approximation of the j-th von Mises.

References

'S. Jung, K. Park, and B. Kim (2021), "Clustering on the torus by conformal prediction"

Examples

```
data <- ILE[1:200, 1:2]

EMsinvMmix(data, J = 3,
           THRESHOLD = 1e-10, maxiter = 200,
           type = "general", kmax = 500, verbose = FALSE)
```

---

grid.torus *Grid on torus*

Description

`grid.torus` returns an equally-spaced grid on torus.

Usage

```
grid.torus(d = 2, grid.size = 100)
```

Arguments

`d`	number for dimension. Default is 2.
`grid.size`	number of grid for each axis. Default value is 100.

Value

returns (grid.size) x (grid.size) numeric matrix which indicates the grid points on torus.

Examples

```
grid.torus(d = 2, grid.size = 100)
```

hyperparam.alpha *Selecting optimal level based on the runs of the number of clusters*

Description

hyperparam.alpha evaluates the numbers of clusters for various levels, and select the optimal level based on the runs of the cluster numbers.

Usage

```
hyperparam.alpha(icp.torus, alphavec = NULL, alpha.lim = 0.15)
```

Arguments

icp.torus	an object containing all values to compute the conformity score, which will be constructed with icp.torus.score.
alphavec	either a scalar or a vector, or even NULL for the levels. Default value is NULL. If NULL, then alphavec is automatically generated as a sequence from 0 to alpha.lim.
alpha.lim	a positive number lower than 1, which is the upper bound of Default is 0.15.

Value

returns a list object which contains a data.frame for the numbers of clusters corresponding to the levels and the optimal level.

See Also

hyperparam.J, hyperparam.torus icp.torus.score

Examples

```
data <- toydata2[, 1:2]
n <- nrow(data)
split.id <- rep(2, n)
split.id[sample(n, floor(n/2))] <- 1
icp.torus <- icp.torus.score(data, split.id = split.id, method = "kmeans",
                             kmeansfitmethod = "ge", init = "h",
                             param = list(J = 25), verbose = TRUE)
hyperparam.alpha(icp.torus)
```

`hyperparam.J` *Selecting optimal number of mixture components based on various criteria*

Description

`hyperparam.J` evaluates criterion for each `icp.torus` objects, and select the optimal number of mixture components based on the evaluated criterion.

Usage

```
hyperparam.J(data, icp.torus.objects, option = c("risk", "AIC", "BIC"))
```

Arguments

`data`	n x d matrix of toroidal data on $[0, 2\pi)^d$ or $[-\pi, \pi)^d$
`icp.torus.objects`	a list whose elements are icp.torus objects, generated by `icp.torus.score`.
`option`	a string one of "risk", "AIC", or "BIC", which determines the criterion for the model selection. "risk" is based on the negative log-likelihood, "AIC" for the Akaike Information Criterion, and "BIC" for the Bayesian Information Criterion.

Value

returns a list object which contains a `data.frame` for the evaluated criterion corresponding to each number of components, the optimal number of components, and the corresponding `icp.torus` object.

References

Akaike (1974), "A new look at the statistical model identification", Schwarz, Gideon E. (1978), "Estimating the dimension of a model"

See Also

`icp.torus.score`, `hyperparam.torus`, `hyperparam.alpha`

Examples

```
data <- toydata2[, 1:2]
n <- nrow(data)
split.id <- rep(2, n)
split.id[sample(n, floor(n/2))] <- 1
Jvec <- 3:35
icp.torus.objects <- list()
for (j in Jvec){
```

```
  icp.torus.objects[[j]] <- icp.torus.score(data, split.id = split.id, method = "kmeans",
                                            kmeansfitmethod = "ge", init = "h",
                                            param = list(J = j), verbose = TRUE)
}
hyperparam.J(data, icp.torus.objects, option = "risk")
```

`hyperparam.torus` *Selecting optimal hyperparameters for the conformal prediction set*

Description

`hyperparam.torus` selects optimal hyperparamters for constructing the conformal prediction set, based on the type of postulated model and the criterion.

Usage

```
hyperparam.torus(
  data,
  icp.torus.objects = NULL,
  option = c("elbow", "risk", "AIC", "BIC"),
  split.id = NULL,
  Jvec = 3:35,
  kvec = 20:100,
  alphavec = NULL,
  alpha.lim = 0.15,
  method = c("kde", "mixture", "kmeans"),
  mixturefitmethod = c("circular", "axis-aligned", "general", "Bayesian"),
 kmeansfitmethod = c("homogeneous-circular", "heterogeneous-circular", "ellipsoids",
    "general"),
  init = c("kmeans", "hierarchical"),
  eval.point = NULL,
  additional.condition = TRUE,
  kmax = 500,
  THRESHOLD = 1e-10,
  maxiter = 200,
  verbose = FALSE
)
```

Arguments

`data`	n x d matrix of toroidal data on $[0, 2\pi)^d$ or $[-\pi, \pi)^d$
`icp.torus.objects`	list whose elements are icp.torus objects, generated by `icp.torus.score`
`option`	A string. One of "elbow", "risk", "AIC", or "BIC", which determines the criterion for the model selection. "risk" is based on the negative log-likelihood, "AIC" for the Akaike Information Criterion, and "BIC" for the Bayesian Infor-

mation Criterion. "elbow" is based on minimizing the criterion used in Jung, et. al.(2021).

`split.id` a n-dimensinal vector consisting of values 1 (estimation) and 2(evaluation)

`Jvec` either a scalar or a vector for the number of mixture components. Default value is 3:35.

`kvec` either a scalar or a vector for the concentration parameter. Default value is 20:100.

`alphavec` either a scalar or a vector, or even `NULL` for the levels. Default value is `NULL`. If `NULL`, then `alphavec` is automatically generated as a sequence from 0 to `alpha.lim`.

`alpha.lim` a positive number lower than 1, which is the upper bound of Default is 0.15.

`method` A string. One of "all", "kde", "mixture", and "kmeans" which determines the model or estimation methods. If "kde", the model is based on the kernel density estimates. It supports the kde-based conformity score only. If "mixutre", the model is based on the von Mises mixture, fitted with an EM algorithm. It supports the von Mises mixture and its variants based conformity scores. If "kmeans", the model is also based on the von Mises mixture, but the parameter estimation is implemented with the elliptical k-means algorithm illustrated in Appendix. It supports the log-max-mixture based conformity score only. Default is "all". If the dimension of data space is greater than 2, only "kmeans" is supported.

`mixturefitmethod`
A string. One of "circular", "axis-aligned", and "general" which determines the constraint of the EM fitting. Default is "axis-aligned". This argument only works for `method = "mixture"`.

`kmeansfitmethod`
A string. One of "general", ellipsoids", "heterogeneous-circular" or "homogeneous-circular". If "general", the elliptical k-means algorithm with no constraint is used. If "ellipsoids", only the one iteration of the algorithm is used. If"heterogeneous-circular", the same as above, but with the constraint that ellipsoids must be spheres. If "homogeneous-circular", the same as above but the radii of the spheres are identical. This argument only works for method = "kmeans".

`init` determine the initial parameter of "kmeans" method, for option "general". Must be "kmeans" or "hierarchical". If "kmeans", the initial parameters are obtained with extrinsic kmeans method. If "hierarchical", the initial parameters are obtained with hierarchical clustering method. Default is "kmeans".

`eval.point` N x N numeric matrix on $[0, 2\pi)^2$. Default input is `grid.torus`.

`additional.condition`
boolean index. If `TRUE`, a singular matrix will be altered to the scaled identity.

`kmax` the maximal number of kappa. If estimated kappa is larger than `kmax`, then put kappa as `kmax`.

`THRESHOLD` number for difference between updating and updated parameters. Default is 1e-10.

`maxiter` the maximal number of iteration. Default is 200.

verbose — boolean index, which indicates whether display additional details as to what the algorithm is doing or how many loops are done. Moreover, if `additional.condition` is `TRUE`, the warning message will be reported.

Value

returns a list object which contains `data.frame` objects for the evaluated criterion corresponding to each hyperparameter, selected hyperparameters based on the designated criterion, and an `icp.torus` object based the selected hyperparameters.

References

S. Jung, K. Park, and B. Kim (2021), "Clustering on the torus by conformal prediction", Akaike (1974), "A new look at the statistical model identification", Schwarz, Gideon E. (1978), "Estimating the dimension of a model"

Examples

```
data <- toydata2[, 1:2]
n <- nrow(data)
split.id <- rep(2, n)
split.id[sample(n, floor(n/2))] <- 1
Jvec <- 3:35
icp.torus.objects <- list()
for (j in Jvec){
  icp.torus.objects[[j]] <- icp.torus.score(data, split.id = split.id, method = "kmeans",
                                            kmeansfitmethod = "ge", init = "h",
                                            param = list(J = j), verbose = TRUE)
}
hyperparam.torus(data, icp.torus.objects, option = "risk")
```

---

icp.torus.eval *Inductive prediction sets for each level*

Description

`icp.torus.eval` evaluates whether each pre-specified evaluation point is contained in the inductive conformal prediction sets for each given level.

Usage

```
icp.torus.eval(icp.torus, level = 0.1, eval.point = grid.torus())
```

Arguments

icp.torus	an object containing all values to compute the conformity score, which will be constructed with icp.torus.score.
level	either a scalar or a vector, or even NULL. Default value is 0.1.
eval.point	N x N numeric matrix on $[0, 2\pi)^2$. Default input is grid.torus.

Value

returns a cp object with the boolean values which indicate whether each evaluation point is contained in the inductive conformal prediction sets for each given level.

References

S. Jung, K. Park, and B. Kim (2021), "Clustering on the torus by conformal prediction"

See Also

grid.torus, icp.torus.score

Examples

```
data <- toydata1[, 1:2]

icp.torus <- icp.torus.score(data, method = "all",
                             mixturefitmethod = "general",
                             param = list(J = 4, concentration = 25))

icp.torus.eval(icp.torus, level = c(0.1, 0.08), eval.point = grid.torus())
```

---

icp.torus.score *Conformity score for inductive prediction sets*

Description

icp.torus.score prepares all values for computing the conformity score for specified methods.

Usage

```
icp.torus.score(
  data,
  split.id = NULL,
  method = c("all", "kde", "mixture", "kmeans"),
  mixturefitmethod = c("circular", "axis-aligned", "general", "Bayesian"),
 kmeansfitmethod = c("homogeneous-circular", "heterogeneous-circular", "ellipsoids",
    "general"),
```

```
  init = c("kmeans", "hierarchical"),
  additional.condition = TRUE,
  param = list(J = 4, concentration = 25),
  kmax = 500,
  THRESHOLD = 1e-10,
  maxiter = 200,
  verbose = TRUE
)
```

Arguments

`data`	n x d matrix of toroidal data on $[0, 2\pi)^d$ or $[-\pi, \pi)^d$
`split.id`	a n-dimensinal vector consisting of values 1 (estimation) and 2(evaluation)
`method`	A string. One of "all", "kde", "mixture", and "kmeans" which determines the model or estimation methods. If "kde", the model is based on the kernel density estimates. It supports the kde-based conformity score only. If "mixutre", the model is based on the von Mises mixture, fitted with an EM algorithm. It supports the von Mises mixture and its variants based conformity scores. If "kmeans", the model is also based on the von Mises mixture, but the parameter estimation is implemented with the elliptical k-means algorithm illustrated in Appendix. It supports the log-max-mixture based conformity score only. Default is "all". If the dimension of data space is greater than 2, only "kmeans" is supported.
`mixturefitmethod`	A string. One of "circular", "axis-aligned", and "general" which determines the constraint of the EM fitting. Default is "axis-aligned". This argument only works for `method = "mixture"`.
`kmeansfitmethod`	A string. One of "general", ellipsoids", "heterogeneous-circular" or "homogeneous-circular". If "general", the elliptical k-means algorithm with no constraint is used. If "ellipsoids", only the one iteration of the algorithm is used. If"heterogeneous-circular", the same as above, but with the constraint that ellipsoids must be spheres. If "homogeneous-circular", the same as above but the radii of the spheres are identical. This argument only works for method = "kmeans".
`init`	determine the initial parameter of "kmeans" method, for option "general". Must be "kmeans" or "hierarchical". If "kmeans", the initial parameters are obtained with extrinsic kmeans method. If "hierarchical", the initial parameters are obtained with hierarchical clustering method. Default is "kmeans".
`additional.condition`	boolean index. If `TRUE`, a singular matrix will be altered to the scaled identity.
`param`	the number of components for mixture fitting and the concentration parameter in the form of `list(J=j,concentration=k)`.
`kmax`	the maximal number of kappa. If estimated kappa is larger than `kmax`, then put kappa as `kmax`.
`THRESHOLD`	number for difference between updating and updated parameters. Default is 1e-10.

maxiter	the maximal number of iteration. Default is 200.
verbose	boolean index, which indicates whether display additional details as to what the algorithm is doing or how many loops are done. Moreover, if additional.condition is TRUE, the warning message will be reported.

Value

returns an icp.torus object, containing all values to compute the conformity score.

References

S. Jung, K. Park, and B. Kim (2021), "Clustering on the torus by conformal prediction"

Examples

```
data <- toydata1[, 1:2]

icp.torus <- icp.torus.score(data, method = "all",
                             mixturefitmethod = "general",
                             kmeansfitmethod = "general",
                             param = list(J = 4, concentration = 25))
```

ILE *ILE: Structure of the Isoleucine*

Description

An isomer of leucine, essential branched-chain aliphatic amino acid found in many proteins.

Usage

```
ILE
```

Format

This list contains the following components:

tbl a numeric matrix of psi, phi, and chi torsion angles.

Details

ILE data is generated with collection of different pdb files. To select adequate protein data, we use PISCES server. (the method is introduced in articles of references.) To select high-quality protein data, we use several benchmarks: resolution : 1.6A(angstrom) or better, R-factor : 0.22 or better, Sequence percentage identity: <= 25 Then, we select ILE only angular data for each protein data. To see the detail code, visit https://github.com/sungkyujung/ClusTorus

Source

This data is extracted from PISCES server http://dunbrack.fccc.edu/pisces/

References

Data description is from https://www.rcsb.org/ligand/ILE.

The data extracting method is from Harder, T., Boomsma, W., Paluszewski, M. et al.(2010) "Beyond rotamers: a generative, probabilistic model of side chains in proteins". BMC Bioinformatics 11, 306 doi: 10.1186/1471210511306 and

Kanti V. Mardia , John T. Kent , Zhengzheng Zhang , Charles C. Taylor & Thomas Hamelryck (2012) "Mixtures of concentrated multivariate sine distributions with applications to bioinformatics", Journal of Applied Statistics, 39:11, 2475-2492, DOI: 10.1080/02664763.2012.719221

See Also

Description of the angluar information is from the 'value' part of `torsion.pdb` in the package `bio3d`.

`kde.torus` *Kernel density estimation using circular von Mises distribution*

Description

`kde.torus` returns a kde using independent multivariate von mises kernel.

Usage

```
kde.torus(data, eval.point = NULL, concentration = 25)
```

Arguments

`data`	n x d matrix of toroidal data on $[0, 2\pi)^d$
`eval.point`	N x N numeric matrix on $[0, 2\pi)^d$. Default input is `NULL`, which represents the fine grid points on $[0, 2\pi)^d$.
`concentration`	positive number which has the role of κ of von Mises distribution. Default value is 25.

Value

`kde.torus` returns N-dimensional vector of kdes evaluated at eval.point

References

S. Jung, K. Park, and B. Kim (2021), "Clustering on the torus by conformal prediction"

See Also

grid.torus

Examples

```
data <- ILE[1:200, 1:2]

kde.torus(data)
```

---

kmeans.kspheres *K-Means Clustering to K-Spheres Clustering on Torus*

Description

kmeans.kspheres prepares the parameters for conformity scores which are derived by k-means clustering on torus.

Usage

```
kmeans.kspheres(
  data,
  centers = 10,
 type = c("homogeneous-circular", "heterogeneous-circular", "ellipsoids", "general"),
  init = c("kmeans", "hierarchical"),
  additional.condition = TRUE,
  THRESHOLD = 1e-10,
  maxiter = 200,
  verbose = TRUE
)
```

Arguments

data	data n x d matrix of toroidal data on $[0, 2\pi)^d$
centers	either the number of clusters or a set of initial cluster centers. If a number, a random set of row in x is chosen as the initial centers.
type	character which must be "homogeneous-circular", "heterogeneous-circular", or "general". If "homogeneous-circular", the radii of k-spheres are identical. If "heterogeneous-circular", the radii of k-spheres may be different. If "ellipsoids", cluster with k-ellipsoids without optimized parameters. If, "general", clustering with k-ellipsoids. The parameters to construct the ellipses are optimized with elliptical k-means algorithm, which is modified for toroidal space. See references for the detail. Default is "homogeneous-circular".
init	determine the initial parameter for option "general". Must be "kmeans" or "hierarchical". If "kmeans", the initial parameters are obtained with extrinsic kmeans method. If "hierarchical", the initial parameters are obtained with hierarchical clustering method. Default is "hierarchical".

additional.condition
: boolean index. If TRUE, a singular matrix will be altered to the scalar identity.

THRESHOLD
: number of threshold for difference between updating and updated parameters. Default is 1e-10.

maxiter
: the maximal number of iteration. Default is 200.

verbose
: boolean index, which indicates whether display additional details as to what the algorithm is doing or how many loops are done. Default is TRUE.

Value

returns a sphere.param object, containing all values which determines the shape and location of spheres.

References

S. Jung, K. Park, and B. Kim (2021), "Clustering on the torus by conformal prediction", and Jaehyeok Shin, Alessandro Rinaldo and Larry Wasserman (2019), "Predictive Clustering"

See Also

kmeans.torus

Examples

```
data <- ILE[1:200, 1:2]

kmeans.kspheres(data, centers = 3, type = "general")
```

---

kmeans.torus *K-Means Clustering on Torus*

Description

kmeans.torus implements extrinsic k-means clustering on toroidal space.

Usage

```
kmeans.torus(data, centers = 10, iter.max = 100, nstart = 1)
```

Arguments

data
: n x d matrix of toroidal data on $[0, 2\pi)^d$

centers
: either the number of clusters or a set of initial cluster centers. If a number, a random set of row in x is chosen as the initial centers.

iter.max
: the maximum number of iterations

nstart
: if centers is a number, how many random sets should be chosen?

Details

In Euclidean space, we know that the total sum of squares is equal to the summation of the within cluster sum of squares and the between cluster centers sum of squares. However, toroidal space does not satisfy the property; the equality does not hold. Thus, you need to be careful to use the sum of squares.

Value

returns a `kmeans` object, which contains input data, cluster centers on torus, membership, total sum of squares, within cluster sum of squares, between cluster centers sum of squares, and the size of each cluster.

References

'S. Jung, K. Park, and B. Kim (2021), "Clustering on the torus by conformal prediction"

See Also

`kmeans`, `ang.minus`

Examples

```
data <- ILE[1:200, 1:2]

kmeans.torus(data, centers = 2,
             iter.max = 100, nstart = 1)
```

`on.torus` *Transform the angular data to be on principal interval*

Description

`on.torus` transforms d-dimensional angular data to be on $[0, 2\pi)^d$.

Usage

```
on.torus(x)
```

Arguments

`x`	d-dimensional angular data(vector or matrix) whose unit is the radian.

Value

d-dimensional radian-unit angular data on $[0, 2\pi)^d$.

Examples

```
data <- SARS_CoV_2$tbl[1:200, 1:2]
data <- data * pi / 180

on.torus(data)
```

`pred.kmeans.torus` *Prediction for Extrinsic Kmeans Clustering*

Description

`pred.kmeans.torus` predicts the cluster for each data point.

Usage

```
pred.kmeans.torus(data, kmeans)
```

Arguments

`data`	n x d matrix of toroidal data on $[0, 2\pi)^d$.
`kmeans`	a `kmeans` object, which contains input data, cluster centers on torus, membership, total sum of squares, within cluster sum of squares, between cluster centers sum of squares, and the size of each cluster. See `kmeans.torus`

Value

a vector whose elements indicate the labels of predicted clusters.

References

'S. Jung, K. Park, and B. Kim (2021), "Clustering on the torus by conformal prediction"

See Also

`kmeans.torus`

Examples

```
data <- ILE[1:200, 1:2]

split.id <- sample(1:2, nrow(data), replace = TRUE)
data.train <- data[split.id == 1, ]
data.test <- data[split.id == 2, ]

kmeans <- kmeans.torus(data.train, centers = 2,
             iter.max = 100, nstart = 1)

pred.kmeans.torus(data.test, kmeans)
```

SARS_CoV_2 *6VXX: Structure of the SARS-CoV-2 spike glycoprotein(closed state)*

Description

The torsion angle dataset of the SARS-CoV-2 spike glycopreotein.

Usage

```
SARS_CoV_2
```

Format

This list contains the following components:

`psi` protein backbone chain angle for atoms C, N, C_alpha and C, in arc-degree.

`phi` protein backbone chain angle for atoms N, C_alpha, C and N, in arc-degree.

`omega` protein backbone chain angle for atoms C_alpha, C, N and C-alpha, in arc-degree.

`chi1` side chain torsion angle for atoms N, C_alpha, C_beta and *G, in arc-degree.

`chi2` side chain torsion angle for atoms C_alpha, C_beta, *G and *D, in arc-degree.

`chi3` side chain torsion angle for atomsC_beta, *G, *D and *E, in arc-degree.

`chi4` side chain torsion angle for atoms *G, *D, *E and *Z, in arc-degree.

`chi5` side chain torsion angle for atoms *D, *E, *Z and NH1, in arc-degree.

`alpha` virtual torsion angle between consecutive C_alpha atoms.

`coords` numeric matrix of 'justified' coordinates.

`tbl` a numeric matrix of psi, phi, and chi torsion angles.

Source

This data can be downloaded in https://www.rcsb.org/structure/6VXX, or with using R package `bio3d`.

References

Walls, A.C., et al. (2020), "Structure of the SARS-CoV-2 spike glycoprotein (closed state)" Cell 181: 281, DOI:10.2210/pdb6vxx/pdb. Retrived from https://www.wwpdb.org/pdb?id=pdb_00006vxx

See Also

Description of the angluar information is from the 'value' part of `torsion.pdb` in the package `bio3d`.

tor.minus *Toroidal subtraction*

Description

tor.minus computes angular subtraction bewtween n x d toroidal data and a d dimensional vector.

Usage

```
tor.minus(data, mu)
```

Arguments

data	n x d matrix of toroidal data
mu	a d-dimensinal vector

Value

angular subtraction bewtween n x d toroidal data and a d dimensional vector.

References

S. Jung, K. Park, and B. Kim (2021), "Clustering on the torus by conformal prediction"

See Also

ang.minus

Examples

```
data <- ILE[1:200, 1:2]
Mu1 <- c(4.5, 3)
tor.minus(data, Mu1)
```

toydata1 *toydata1: Labelled Data for 5 Clusters*

Description

Artificially generated data on the 2 dimensional torus

Usage

```
toydata1
```

Format

This `data.frame` contains the following components:

`phi` column for the first angle

`psi` column for the second angle

`label` column for the clustering membership

Details

toydata1 is an artificial data generated from a mixture of 5 clusters, where three clusters are sampled from bivariate normal distributions and the other two are each sampled from the uniform distribution on a rectangle.

References

This simulation data is from S. Jung, K. Park, B. Kim (2021) "Clustering on the torus by conformal prediction". Annals of Applied Statistics

---

`toydata2` *toydata2: Labelled Data for 3 Clusters*

Description

Artificially generated data on the 2 dimensional torus

Usage

```
toydata2
```

Format

This `data.frame` contains the following components:

`phi` column for the first angle

`psi` column for the second angle

`label` column for the clustering membership

Details

toydata2 is an artificial data generated from a mixture of 3 clusters, where the first cluster is sampled from a spherical normal distribution, the second cluster is from the uniform distribution on a large "L"-shaped region, and the third cluster of size 50 is sampled from the uniform distribution on the entire 2-dimensional torus.

References

This simulation data is from S. Jung, K. Park, B. Kim (2021) "Clustering on the torus by conformal prediction". Annals of Applied Statistics

`wtd.stat.ang` *Weighted extrinsic mean direction and mean resultant length*

Description

`wtd.stat.ang` computes weighted extrinsic mean direction and mean resultant length.

Usage

```
wtd.stat.ang(data, w)
```

Arguments

`data`	angular data whose elements are in $[0, 2\pi)$
`w`	numeric vector whose each element is non-negative and `sum(w) == 1`. Moreover, the length of `w` is the same with `nrow(data)`.

Value

list which is consisting of the following components:

`Mean` weighted extrinsic mean direction

`R` mean resultant length

References

S. Jung, K. Park, and B. Kim (2021), "Clustering on the torus by conformal prediction"

Examples

```
data <- matrix(c(pi/3, pi/3, pi/2,
                 pi, pi/4, pi/2,
                 0, pi/3, pi/6),
               ncol = 3, byrow = TRUE)
w <- c(0.3, 0.3, 0.4)
wtd.stat.ang(data, w)
```

Index